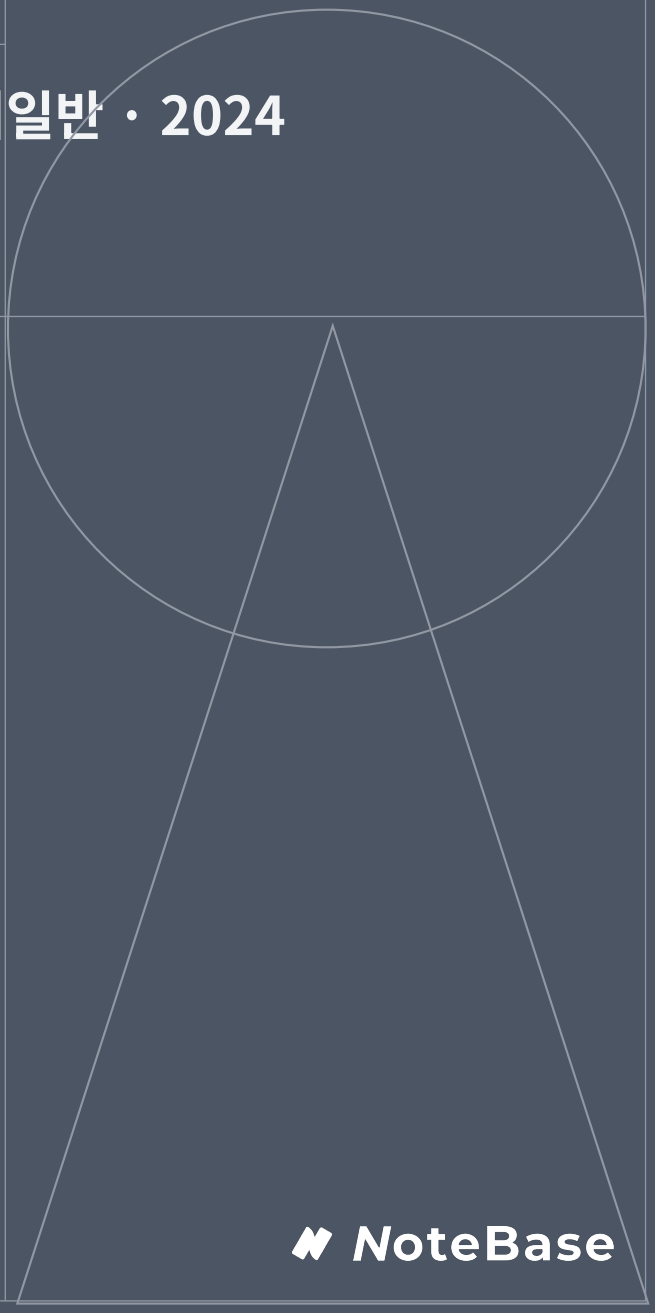
기출OX 해설집

공무원/법원직/9급 · 컴퓨터일반 · 2024



NoteBase

기출OX 해설집

공무원/법원직/9급 · 컴퓨터일반 · 2024

공무원/법원직/9급 · 컴퓨터일반 · 2024

문 1. 다음 C 언어 프로그램의 출력 결과는?

```
#include <stdio.h>
int f(int *a)
{    static int b=0;
     (*a)++; b=b+2;
     return b;
}
int main()
{    int x=0, y=0;
     x = f(&y);
     y = y + f(&x);
     printf("%d", x+y);
}
```

① 5

② 6

③ 7

④ 8

정답 4번

출제 포인트

포인터로 전달된 변수의 증가와 정적 지역 변수의 값 유지 과정을 함수 호출 순서대로 추적하여 $x + y$를 구하는 문제이다.

NoteBase

조건 정리

초기값은 $x = 0, y = 0$이다. 함수 f는 포인터가 가리키는 값을 1 증가시키고, 정적 지역 변수 b를 2 증가시킨 뒤 b를 반환한다. 정적 지역 변수 b는 함수 호출이 끝나도 값이 유지된다.

계산 과정

1. 첫 번째 호출 x=f(\&y)에서 y는 0에서 1이 되고, b는 0에서 2가 되어 2를 반환하므로 $x = 2$가 된다.
2. 두 번째 호출이 포함된 y=y+f(\&x)에서 기존 y는 1이다. 함수 호출로 x는 2에서 3이 되고, 유지되던 b는 2에서 4가 되어 4를 반환한다. 따라서 $y = 1 + 4 = 5$가 된다.
3. 출력값은 $x + y = 3 + 5 = 8$이다.

선지 판단

① X — 출력값은 5가 아니라 8이다. 올바르게는 두 함수 호출 후 $x = 3, y = 5$이므로 8이 출력된다.
② X — 출력값은 6이 아니라 8이다. 두 번째 호출에서도 정적 변수 b가 누적되어 4를 반환한다는 점을 반영해야 한다.
③ X — 출력값은 7이 아니라 8이다. 두 번째 f(\&x) 호출이 x 자체를 2에서 3으로 증가시키므로 최종 합은 8이다.
④ O — 최종적으로 $x = 3, y = 5$이므로 *printf*는 8을 출력한다.

오답 포인트

정적 지역 변수 b를 호출할 때마다 0으로 초기화한다고 보거나, 포인터를 통한 (*a) + +가 실제 변수 x와 y를 바꾼다는 점을 놓치기 쉽다. 특히 두 번째 호출은 반환값 4만 더하는 것이 아니라 x도 3으로 변경한다.

다시 나오면 이렇게 판단

함수 호출마다 포인터가 가리키는 변수의 변화, 정적 변수의 누적값, 함수 반환값을 따로 기록한 뒤 대입문을 완성하고 마지막 출력식을 계산한다.

암기 포인트

포인터 매개변수는 원본 변수를 바꾸고, *static* 지역 변수는 호출 사이에도 값을 유지한다.

NoteBase

문 2. 다음 빈칸에 들어갈 용어는 무엇인가?

C 언어로 작성한 프로그램(“A”라고 칭함)과 스크립트 언어(예: shell script)로 작성한 프로그램(“B”라고 칭함)은 실행 파일 형태로 만들어진 후 실행될 때 그 실행 방식에 차이가 있다. A의 경우 실행파일이 기계어를 담고 있으므로 메모리에 직접 로드(load) 되어 실행되는 반면 B의 경우 실행파일에 기계어가 아닌 텍스트 파일이 저장돼 있으므로 직접 실행되지 못하고 [_____]가 먼저 실행되면서 B의 내용을 한 줄씩 해석하며 실행한다.

① 컴파일러(Compiler)

② 인터프리터(Interpreter)

③ 링커(Linker)

④ 로더(Loader)

정답 2번

출제 포인트

이 문제는 기계어 실행 파일과 스크립트의 실행 방식을 구분하고, 스크립트를 한 줄씩 해석하여 실행하는 프로그램을 묻는다.

핵심 개념

인터프리터는 소스 코드를 실행 시점에 읽고 해석하여 명령을 수행하는 프로그램이다.
컴파일러는 소스 코드 전체를 목적 코드나 기계어로 번역하고, 링커는 목적 파일과 라이브러리를 결합하며, 로더는 완성된 실행 파일을 메모리에 적재한다.
컴파일러는 실행 전 번역, 인터프리터는 실행 중 해석, 링커는 파일 결합, 로더는 메모리 적재가 핵심이다.

선지별 해설

① X — 컴파일러는 C 언어 등의 소스 코드를 목적 코드나 기계어로 번역하는 프로그램이다. ‘스크립트의 내용을 한 줄씩 해석하며 실행한다’는 표현이 틀렸고, 올바르게는 인터프리터가 그 역할을 한다.
② O — 인터프리터는 스크립트의 텍스트 명령을 실행 시점에 차례로 해석하여 수행한다. 셸 스크립트는 셸과 같은 인터프리터가 내용을 읽어 실행하므로 빈칸에 알맞다.
③ X — 링커는 여러 목적 파일과 필요한 라이브러리를 결합하고 외부 참조를 해결하여 실행 파일을 만드는 프로그램이다. ‘스크립트를 한 줄씩 해석하여 실행한다’는 표현이 틀렸고, 올바르게는 인터프리터의 기능이다.
④ X — 로더는 실행 파일을 메모리에 적재하고 실행할 수 있도록 준비하는 프로그램이다. ‘텍스트로 된 스크립트의 내용을 한 줄씩 해석한다’는 표현이 틀렸고, 올바르게는 인터프리터가 해석을 담당한다.

NoteBase

오답 포인트

문장에 '메모리에 직접 로드'라는 표현이 있어 ④ 로더를 고르기 쉽다. 그러나 빈칸 뒤의 결정적 단서는 'B의 내용을 한 줄씩 해석하며 실행한다'이며, 이는 로더가 아니라 인터프리터의 기능이다.

다시 나오면 이렇게 판단

'번역'이면 컴파일러, '목적 파일·라이브러리 결합'이면 링커, '메모리 적재'이면 로더, '실행하면서 한 줄씩 해석'이면 인터프리터로 구분한다.

암기 포인트

컴파일러는 미리 번역하고, 인터프리터는 실행하며 해석한다. 링커는 결합하고, 로더는 적재한다.

NoteBase

문 3. 다음 설명 중 가장 옳지 않은 것은?

① 인증기술의 분류는 사용자 인증기술, 메시지 인증기술, 디바이스 인증기술 등으로 구분한다.

② 인증요소로 소지기반은 인증자와 검증자만 알고 있는 지식을 서로 비교하여 본인임을 인증하는 것이다.

③ 전자인증서는 전자상거래나 비즈니스를 위한 문서교환 시 사용자의 신원과 문서의 내용을 보증하는 문서이다.

④ 생체인식 정보기반은 인증자의 신체적 특성을 이용하여 본인임을 인증하는 것이다.

정답 2번

출제 포인트

인증기술의 분류와 지식·소지·생체 기반 인증요소의 구별을 묻는 문제이다.

핵심 개념

지식기반 인증은 비밀번호나 PIN처럼 사용자가 알고 있는 정보로 본인임을 확인한다. 소지기반 인증은 보안토큰, 스마트카드, 휴대전화처럼 사용자가 가진 인증 수단으로 확인한다. 생체정보기반 인증은 지문, 홍채, 얼굴 등 사용자의 신체적 특성을 이용한다.

선지별 해설

① O — 인증 대상을 기준으로 사용자 인증, 메시지 인증, 디바이스 인증 등으로 구분할 수 있다. 사용자는 접속 주체를, 메시지는 송신자와 내용의 무결성을, 디바이스는 접속 기기의 정당성을 확인한다.
② X — 틀린 표현은 '소지기반은 인증자와 검증자만 알고 있는 지식을 서로 비교'한다는 부분이다. 이는 비밀번호나 PIN 등을 이용하는 지식기반 인증의 설명이다. 소지기반 인증은 보안토큰, 스마트카드, 휴대전화 등 사용자가 소유한 인증 수단을 제시해 본인임을 확인하는 방식이다.
③ O — 전자인증서는 소유자의 신원과 공개키의 결합을 증명한다. 이를 전자서명 검증에 사용하여 서명자의 신원과 문서 내용의 진정성·무결성을 확인할 수 있다.
④ O — 생체정보기반 인증은 지문, 홍채, 얼굴과 같은 인증자의 신체적 특성을 이용하여 본인 여부를 확인하는 방식이다.

오답 포인트

②는 '소지기반'이라는 이름을 붙여 놓고 실제 정의는 '알고 있는 것'을 사용하는 지식기반 인증으로 바꾸어 놓았다. 인증요소는 지식, 소지, 생체의 기준어를 중심으로 구별해야 한다.

다시 나오면 이렇게 판단

인증 수단이 '알고 있는 것'이면 지식기반, '가지고 있는 것'이면 소지기반, '신체에 존재하는 것'이면 생체정보기반으로 판단한다.

NoteBase

암기 포인트

비밀번호는 지식, 토큰은 소지, 지문은 생체이다.

NoteBase

문 4. 다음에서 문맥 교환(Context switch)이 반드시 발생하는 경우는? 단, CPU 대기 큐(Ready queue)는 비어있지 않다고 가정한다.

① 프로세스가 요청한 I/O가 완료되어 인터럽트가 발생한 경우

② 라운드 로빈(Round Robin) 방식으로 CPU 스케줄링을 하는 시스템에서 프로세스의 CPU 할당시간(Time quantum)이 만료되었음을 알려주는 인터럽트가 발생한 경우

③ 프로세스가 시스템콜(System call)을 하는 경우

④ 다중 쓰레드(Thread)로 구성된 프로세스 내에서 쓰레드 중 하나가 I/O 요청을 하는 경우

정답 2번

출제 포인트

인터럽트나 시스템 콜로 커널이 실행되는 것과, 실행 중인 프로세스의 상태를 저장하고 다른 프로세스의 상태를 복원하는 문맥 교환을 구별하는 문제이다.

핵심 개념

문맥 교환은 CPU를 사용하던 프로세스 또는 쓰레드의 실행 문맥을 저장하고, 새로 실행할 대상의 문맥을 복원하여 CPU 사용 주체를 바꾸는 작업이다.
인터럽트와 시스템 콜은 커널 진입의 원인이지만, 처리 후 원래 실행 대상으로 복귀할 수도 있으므로 그 자체가 항상 문맥 교환을 뜻하지는 않는다.
라운드 로빈에서는 시간 할당량이 끝나면 실행 중인 프로세스를 준비 큐로 보내고 다음 준비 프로세스에 CPU를 넘긴다.

NoteBase

선지별 해설

① X — I/O 완료 인터럽트로 대기하던 프로세스가 준비 상태가 되더라도, 스케줄링 정책과 우선순위에 따라 현재 프로세스가 계속 실행될 수 있다. 틀린 표현은 '인터럽트가 발생하면 반드시 문맥 교환이 발생한다'는 취지이며, 올바르게는 I/O 완료 인터럽트가 문맥 교환을 유발할 수 있지만 반드시 유발하는 것은 아니다.
② O — 라운드 로빈에서 시간 할당량 만료 인터럽트가 발생하면 실행 중인 프로세스의 CPU 점유를 끝내고 준비 큐의 다음 프로세스를 실행한다. 문제에서 준비 큐가 비어 있지 않다고 했으므로 CPU 사용 주체가 바뀌어 문맥 교환이 반드시 발생한다.
③ X — 시스템 콜은 사용자 모드에서 커널 모드로 진입하는 동작이지만, 처리가 끝난 뒤 같은 프로세스로 복귀할 수 있다. 틀린 표현은 '시스템 콜을 하면 반드시 문맥 교환이 발생한다'는 취지이며, 올바르게는 시스템 콜이 현재 프로세스를 블록시키거나 스케줄링을 일으킬 때 문맥 교환이 발생할 수 있다.
④ X — 한 쓰레드의 I/O 요청이 비동기·비차단 방식일 수도 있고, 쓰레드 구현 및 I/O 처리 방식에 따라 그 쓰레드나 프로세스가 반드시 CPU를 내놓는 것은 아니다. 틀린 표현은 '다중 쓰레드 중 하나가 I/O를 요청하면 반드시 문맥 교환이 발생한다'는 취지이며, 올바르게는 해당 쓰레드가 차단되어 스케줄러가 다른 실행 대상을 선택할 때 문맥 교환이 발생한다.

오답 포인트

인터럽트 또는 시스템 콜이 발생하면 곧바로 문맥 교환이 일어난다고 오해하기 쉽다. 핵심 함정은 '발생할 수 있다'와 '반드시 발생한다'의 차이이다. ②는 시간 할당량 만료와 준비 큐가 비어 있지 않다는 조건이 함께 주어져 다음 실행 대상으로의 교체가 보장된다.

다시 나오면 이렇게 판단

사건이 커널 진입만 일으키는지, 아니면 현재 실행 대상의 CPU 사용을 끝내고 다른 준비 대상을 선택하도록 강제하는지를 확인한다. 후자가 조건상 보장될 때 문맥 교환이 반드시 발생한다.

암기 포인트

인터럽트·시스템 콜은 문맥 교환의 계기가 될 수 있을 뿐이고, 라운드 로빈의 시간 할당량 만료와 비어 있지 않은 준비 큐는 실행 대상 교체를 보장한다.

NoteBase

문 5. 아래의 C 언어 프로그램을 컴파일 후 명령행인자(command line argument)로 x y를 입력해서 실행했을 때 파일이 하나 생성된다. 생성된 파일 이름은 무엇인가?

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    FILE *z = fopen(argv[2], "w");
    fprintf(z, "hello");
    fclose(z);
}
```

① w

② x

③ y

④ z

정답 3번

출제 포인트

C 언어의 명령행 인자가 argv 배열에 저장되는 순서와 fopen 함수에서 파일 이름을 지정하는 인수를 적용하는 문제이다.

사례 정리

프로그램 실행 시 명령행인자로 x와 y를 차례로 입력한다.
argv[0]에는 실행 프로그램 이름이, argv[1]에는 x가, argv[2]에는 y가 저장된다.
fopen(argv[2], "w")는 argv[2]가 가리키는 이름의 파일을 쓰기 모드로 연다.

핵심 개념

argv는 명령행에서 전달된 문자열들을 저장하는 배열이다. 첫 원소 argv[0]은 실행 프로그램 이름이므로 사용자가 입력한 첫 번째 인자는 argv[1]부터 시작한다.
fopen의 첫 번째 인수는 파일 이름이고, 두 번째 인수 "w"는 쓰기 모드를 뜻한다. 따라서 argv[2]의 값인 y가 생성되는 파일 이름이다.

NoteBase

적용 기준

명령행 입력값을 argv 인덱스에 대응시킨 뒤, fopen의 첫 번째 인수에 실제로 전달되는 값을 확인한다. 이 코드에서는 argv[2]가 y이므로 y라는 파일이 생성된다.

선지별 해설

① X — w는 fopen의 두 번째 인수인 파일 열기 모드이며 파일 이름이 아니다. 올바르게는 파일 이름이 fopen의 첫 번째 인수 argv[2]의 값인 y이다.
② X — x는 사용자가 입력한 첫 번째 명령행인자로 argv[1]에 저장되지만, 코드가 파일 이름으로 사용하는 것은 argv[2]이다. 올바르게는 두 번째 명령행인자 y가 파일 이름이다.
③ O — y는 두 번째 명령행인자이므로 argv[2]에 저장되고, fopen(argv[2], "w")에 따라 y라는 파일이 생성된다.
④ X — z는 fopen이 반환한 파일 포인터를 저장하는 변수명일 뿐 파일 이름이 아니다. 올바르게는 fopen의 첫 번째 인수 argv[2]가 가리키는 y가 파일 이름이다.

오답 포인트

argv[0]부터 사용자가 입력한 인자가 저장된다고 착각하면 x와 y의 인덱스를 하나씩 잘못 대응하게 된다. 또한 "w"는 파일 이름이 아니라 쓰기 모드이고, z는 파일 포인터 변수라는 점을 구분해야 한다.

다시 나오면 이렇게 판단

먼저 argv[0]을 실행 프로그램 이름으로 놓고 사용자 입력을 argv[1]부터 차례로 대응한다. 그다음 fopen의 첫 번째 인수에서 파일 이름을, 두 번째 인수에서 열기 모드를 확인한다.

암기 포인트

argv[0]은 프로그램 이름, argv[1]부터 사용자 인자이며, fopen(파일 이름, 모드)의 순서이다.

NoteBase

문 6. 다음 설명 중 가장 옳지 않은 것은?

① 명세 기반 테스트를 블랙박스 테스트라하며 방법으로는 신택스 기법, 동등 분할 기법, 경계 값 분석 기법 등이 있다.

② 맥케이브의 기본경로 테스트에서 지역(Region)을 계산한 순환복잡도와 분기(Predicate)를 계산한 순환복잡도의 결과는 일치한다.

③ 회귀 테스트(Regression)는 모듈 회귀 테스트와 증분적 회귀 테스트가 있다.

④ 통합 테스트는 단위 테스트가 끝난 모듈을 통합하는 과정에서 발생하는 오류를 찾는 테스트를 말한다.

정답 3번

출제 포인트

소프트웨어 테스트의 분류와 목적, 맥케이브 순환복잡도 계산법, 회귀 테스트의 유형을 구별하는 문제이다.

핵심 개념

명세 기반 테스트는 프로그램 내부 구조를 보지 않고 입력과 예상 출력 등 명세를 기준으로 검사하는 블랙박스 테스트이다.
기본경로 테스트는 제어 흐름 그래프의 순환복잡도로 독립 경로 수를 정하는 구조 기반 테스트이다.
회귀 테스트는 소프트웨어를 변경한 뒤 기존 기능에 새로운 결함이 생기지 않았는지 다시 확인하는 테스트이다.
대표적으로 명세 변경 여부에 따라 교정적 회귀 테스트와 점진적 회귀 테스트로 구분한다.
통합 테스트는 단위 테스트를 마친 모듈을 결합하면서 인터페이스와 상호작용에서 발생하는 오류를 찾는다.

선지별 해설

① O. 명세를 기준으로 외부 동작을 확인하므로 명세 기반 테스트는 블랙박스 테스트에 해당한다. 신택스 기법, 동등 분할 기법, 경계값 분석 기법은 대표적인 명세 기반 기법이다.
② O. 연결된 제어 흐름 그래프에서 순환복잡도 $V(G)$는 영역 수로 구할 수 있고, 분기 노드 수를 P라 하면 $V(G) = P + 1$로도 구할 수 있으므로 두 결과는 일치한다.
③ X. 틀린 부분은 회귀 테스트의 유형을 '모듈 회귀 테스트와 증분적 회귀 테스트'로 짝지은 것이다. 올바르게는 명세가 바뀌지 않아 기존 테스트 사례를 재사용하는 교정적 회귀 테스트와, 명세 변경에 맞추어 테스트 사례를 수정하거나 추가하는 점진적 회귀 테스트로 구분한다.
④ O. 통합 테스트는 단위 테스트가 끝난 모듈들을 결합한 뒤 모듈 사이의 인터페이스, 데이터 전달, 호출 관계 등 통합 과정에서 생기는 오류를 찾는 테스트이다.

NoteBase

오답 포인트

③의 '모듈'과 '증분적'은 모두 테스트 문맥에서 쓰일 수 있어 자연스러워 보이지만, 회귀 테스트의 대표적인 대응 분류는 교정적과 점진적이다. 또한 ②에서는 영역 수 자체와 분기 노드 수 자체가 같은 것이 아니라, 각각의 공식으로 계산한 순환복잡도 값이 같다는 점을 구별해야 한다.

다시 나오면 이렇게 판단

테스트 기법은 무엇을 기준으로 검사하는지와 어느 단계에서 수행하는지를 나누어 판단한다. 명세 기준이면 블랙박스, 제어 흐름 기준이면 구조 기반, 모듈 결합 단계이면 통합 테스트이다. 회귀 테스트의 고전적 분류는 명세가 그대로이면 교정적, 명세까지 바뀌면 점진적으로 판단한다.

암기 포인트

명세는 블랙박스, 경로는 순환복잡도, 결합은 통합 테스트, 회귀는 교정적·점진적이다.

NoteBase

문 7. 다음 설명 중 가장 옳지 않은 것은?

① 일반적으로 데이터 모델은 데이터 구조, 연산, 제약조건으로 구성된다.

② 논리적 데이터 모델링은 개념세계의 데이터를 데이터베이스에 저장할 구조를 결정하고 이 구조로 표현하는 작업을 의미한다.

③ 제약조건에는 구조적 측면의 제약사항과 연산을 적용하는 경우 허용할 수 있는 의미적 측면의 제약사항이 있다.

④ 개념적 데이터 모델에서 데이터 구조는 데이터를 어떤 모습으로 저장할 것인지를 표현한다.

정답 4번

출제 포인트

데이터 모델의 구성 요소와 개념적·논리적 데이터 모델링의 역할을 구분하는 문제이다.

핵심 개념

데이터 모델은 현실 세계의 데이터를 데이터베이스에서 다룰 수 있도록 구조, 연산, 제약조건으로 표현한 체계이다.
개념적 데이터 모델링은 현실 세계에서 무엇을 관리할지를 개체·속성·관계로 추상화한다. 논리적 데이터 모델링은 개념적 구조를 관계, 속성, 키 등 데이터베이스에 저장할 논리 구조로 변환한다. 실제 파일 배치나 인덱스 같은 저장 방법은 물리적 데이터 모델링에서 결정한다.

선지별 해설

① O — 데이터 모델의 기본 구성 요소는 데이터 구조, 연산, 제약조건이다. 구조는 데이터의 정적 형태를, 연산은 데이터를 처리하는 방법을, 제약조건은 허용되는 데이터 상태와 연산 범위를 나타낸다.
② O — 논리적 데이터 모델링은 개념적 모델을 데이터베이스가 수용할 수 있는 논리 구조로 바꾸어 표현하는 작업이다. 관계형 모델이라면 관계, 속성, 키 등의 구조를 정한다.
③ O — 제약조건에는 데이터 구조 자체가 지켜야 할 구조적 제약과, 데이터 상태나 연산 결과가 의미상 유효하도록 제한하는 의미적 제약이 포함된다.
④ X — 틀린 표현은 '개념적 데이터 모델에서 데이터 구조는 데이터를 어떤 모습으로 저장할 것인지를 표현한다'이다. 개념적 데이터 모델은 저장 모습을 정하는 단계가 아니라 현실 세계의 개체·속성·관계를 추상적으로 표현하는 단계이다. 올바르게는 '논리적 데이터 모델에서 데이터 구조는 데이터를 데이터베이스에 어떤 논리적 형태로 저장할 것인지를 표현한다'이다.

오답 포인트

④는 '개념적'과 '논리적'의 역할을 바꾸어 놓았다. 개념적 모델링의 핵심은 업무 대상과 관계의 추상화이고, 데이터베이스에 저장할 논리 구조를 정하는 것은 논리적 모델링이다.

NoteBase

다시 나오면 이렇게 판단

‘무엇을 관리하는가’를 개체·속성·관계로 파악하면 개념적 모델링, ‘데이터베이스에 어떤 논리 구조로 표현하는가’를 정하면 논리적 모델링, ‘디스크에 실제로 어떻게 배치하는가’를 정하면 물리적 모델링으로 판단한다.

암기 포인트

개념은 무엇, 논리는 어떤 구조, 물리는 실제 저장이다. 데이터 모델의 3요소는 구조·연산·제약조건이다.

NoteBase

문 8. 오류를 검출하는 기능이 없는 디스크에 대해 오류교정코드를 따로 관리하고 오류가 발생하면 오류교정코드를 이용하여 디스크를 복구하는 RAID는?

① RAID 1.

② RAID 2.

③ RAID 3.

④ RAID 5.

정답 2번

출제 포인트

디스크 자체에 오류 검출 기능이 없을 때 별도의 오류교정코드(ECC) 디스크를 사용하는 RAID 단계를 구별하는 문제이다.

핵심 개념

RAID 1은 같은 데이터를 복제하는 미러링 방식이다.
RAID 2는 데이터를 비트 단위로 분산하고 해밍 코드와 같은 오류교정코드를 별도의 디스크에 저장하는 방식이다.
RAID 3은 데이터를 바이트 단위로 분산하고 전용 패리티 디스크 하나를 사용하는 방식이다.
RAID 5는 데이터를 블록 단위로 분산하며 패리티도 여러 디스크에 분산해 저장하는 방식이다.

선지별 해설

① X — RAID 1은 동일한 데이터를 다른 디스크에 복제하는 미러링 방식이다. 틀린 표현은 'RAID 1'이며, 오류교정코드를 별도로 관리하는 방식은 RAID 2이다.
② O — RAID 2는 오류 검출 기능이 없는 디스크를 전제로 비트 단위 스트라이핑을 하고, 해밍 코드와 같은 오류교정코드를 전용 디스크에 따로 저장하여 오류를 검출·교정한다.
③ X — RAID 3은 바이트 단위 스트라이핑과 하나의 전용 패리티 디스크를 사용하는 방식이다. 틀린 표현은 'RAID 3'이며, 별도의 오류교정코드를 이용해 복구하는 방식은 RAID 2이다.
④ X — RAID 5는 블록 단위 스트라이핑을 하면서 패리티를 특정 전용 디스크가 아니라 여러 디스크에 분산하는 방식이다. 틀린 표현은 'RAID 5'이며, 오류교정코드를 별도 디스크에 관리하는 방식은 RAID 2이다.

오답 포인트

RAID 2의 오류교정코드와 RAID 3·5의 패리티를 모두 복구용 정보로만 묶어 보면 혼동하기 쉽다. 이 문항의 결정적 표현은 '오류 검출 기능이 없는 디스크'와 '오류교정코드를 따로 관리'이며, 이는 RAID 2의 해밍 코드 방식이다.

NoteBase

다시 나오면 이렇게 판단

미러링이면 RAID 1, 비트 단위 분산과 별도 오류교정코드이면 RAID 2, 바이트 단위 분산과 전용 패리티이면 RAID 3, 블록 단위 분산과 분산 패리티이면 RAID 5로 판단한다.

암기 포인트

RAID 2는 비트 단위 분산과 해밍 오류교정코드, RAID 3은 전용 패리티, RAID 5는 분산 패리티이다.

NoteBase

문 9. 리눅스의 파일시스템에서 “chmod 531 A”명령어를 수행한 후 “chmod go-x A”명령어를 수행했을 때 파일 A의 소유자에게는 어떤 접근 권한이 있는가?

① 읽기(read) 권한은 있으나 쓰기(write)와 실행(execution) 권한은 없다.
② 읽기(read)와 쓰기(write) 권한은 있으나 실행(execution) 권한은 없다.
③ 읽기(read)와 실행(execution) 권한은 있으나 쓰기(write) 권한은 없다.
④ 쓰기(write)와 실행(execution) 권한은 있으나 읽기(read) 권한은 없다.

정답 3번

출제 포인트

숫자 모드로 설정한 소유자·그룹·기타 사용자의 권한을 해석한 뒤, 기호 모드가 어느 대상의 권한만 변경하는지 판단하는 문제이다.

조건 정리

숫자 권한은 읽기 4, 쓰기 2, 실행 1의 합으로 나타낸다. chmod 531 A에서 세 자리는 차례로 소유자 5, 그룹 3, 기타 사용자 1의 권한이다. 이어지는 go-x에서 g는 그룹, o는 기타 사용자, -x는 실행 권한 제거를 뜻하므로 소유자 권한은 변경되지 않는다.

사용할 공식

숫자 모드의 권한값은 읽기 4, 쓰기 2, 실행 1을 합산한다. 따라서 소유자의 5는 $5 = 4 + 1$이므로 읽기와 실행 권한을 뜻한다. 이 문제는 첫 번째 숫자가 소유자의 권한을 나타내고, 뒤 명령의 변경 대상에 소유자 u가 포함되지 않기 때문에 이 원리를 사용한다.

계산 과정

1. chmod 531 A를 적용하면 소유자는 5이므로 읽기와 실행 권한을 가지며 쓰기 권한은 없다.
2. 그룹의 3은 쓰기와 실행, 기타 사용자의 1은 실행 권한을 뜻한다.
3. chmod go-x A는 그룹과 기타 사용자에게서 실행 권한만 제거한다. 소유자는 대상이 아니므로 소유자의 권한은 그대로 읽기와 실행이다.

NoteBase

선지 판단

① X — ‘실행 권한은 없다’가 틀렸다. 소유자의 숫자 5에는 실행값 1이 포함되며 go-x도 소유자에게 적용되지 않는다. 올바르게는 읽기와 실행 권한은 있으나 쓰기 권한은 없다.
② X — ‘쓰기 권한은 있다’와 ‘실행 권한은 없다’가 모두 틀렸다. 숫자 5는 읽기 4와 실행 1의 합이므로, 올바르게는 읽기와 실행 권한은 있으나 쓰기 권한은 없다.
③ O — 소유자의 권한 5는 읽기와 실행이고 쓰기는 아니다. 뒤의 go-x는 그룹과 기타 사용자만 바꾸므로 이 권한이 유지된다.
④ X — ‘쓰기와 실행 권한은 있으나 읽기 권한은 없다’가 틀렸다. 이는 소유자의 숫자 5를 잘못 해석한 것이다. 올바르게는 읽기와 실행 권한은 있으나 쓰기 권한은 없다.

오답 포인트

go-x의 go를 모든 사용자로 오해하면 소유자의 실행 권한도 제거된다고 판단하기 쉽다. 그러나 g와 o는 각각 그룹과 기타 사용자이고, 소유자는 u로 표시한다. 또한 숫자 5를 읽기와 쓰기로 착각하기 쉽지만 읽기와 쓰기의 합은 6이고, 5는 읽기와 실행의 합이다.

다시 나오면 이렇게 판단

먼저 세 자리 숫자를 소유자·그룹·기타 사용자 순서로 분리하고, 각 숫자를 4·2·1의 합으로 해석한다. 다음 명령에서는 u, g, o 중 실제 변경 대상을 확인한 뒤 해당 대상의 권한만 더하거나 뺀다.

암기 포인트

권한값은 읽기 4, 쓰기 2, 실행 1이고, 대상은 u 소유자, g 그룹, o 기타 사용자이다. 5는 읽기와 실행이다.

NoteBase

문 10. 다음 파이썬 프로그램의 실행 결과는?

```
def cals(nums):
    if nums == 1 :
        return 1
    else:
        return nums * cals(nums-1)

in_value = 4
result = cals(in_value)
print(result)
```

① 10

② 14

③ 20

④ 24

정답 4번

출제 포인트

재귀 함수의 종료 조건과 반환식을 따라가며 최종 반환값을 계산하는 문제이다.

조건 정리

주어진 값: in_value는 4이다.
구해야 할 값: cals(4)의 반환값이다.
주의할 조건: nums가 1이면 1을 반환하고, 그보다 크면 nums와 cals(nums-1)의 반환값을 곱한다.

사용할 원리

cals(nums)는 nums가 1이 될 때까지 1씩 줄여 재귀 호출한 뒤 각 nums를 차례로 곱한다. 따라서 cals(4)는 4부터 1까지의 곱으로 계산된다.

NoteBase

계산 과정

$cals(1) = 1$
$cals(2) = 2 \times cals(1) = 2$
$cals(3) = 3 \times cals(2) = 6$
$cals(4) = 4 \times cals(3) = 24$
따라서 result에는 24가 저장되고 print(result)는 24를 출력한다.

선지별 해설

① X — 10은 4부터 1까지 더한 값으로, 이 함수의 실행 결과가 아니다. 반환식은 덧셈이 아니라 곱셈이므로 올바른 출력값은 24이다.
② X — 14는 재귀 반환식 nums × cals(nums-1)을 순서대로 적용한 값이 아니다. 올바르게 계산하면 4 × 3 × 2 × 1 = 24이다.
③ X — 20은 마지막 재귀 호출까지 모두 곱한 값이 아니다. 종료 조건 cals(1) = 1을 포함해 반환값을 역으로 곱하면 24이다.
④ O — cals(4) = 4 × 3 × 2 × 1 = 24이므로 프로그램은 24를 출력한다.

오답 포인트

①의 10은 4 + 3 + 2 + 1로 계산했을 때 나오는 값이다. 반환식의 연산자가 +가 아니라 ×라는 점과, nums == 1일 때 재귀 호출이 종료된다는 점을 함께 확인해야 한다.

다시 나오면 이렇게 판단

재귀 함수는 먼저 종료 조건의 반환값을 확정한 뒤, 호출이 되돌아오는 순서대로 반환식을 적용해 최종 값을 계산한다.

암기 포인트

종료 조건부터 역추적한다. 이 함수는 4 × 3 × 2 × 1을 계산하는 팩토리얼 구조이다.

NoteBase

문 11. 다음 설명 중 가장 옳지 않은 것은?

① 임계 영역 문제를 해결하기 위해서는 상호배제, 진행, 제한된 대기 조건을 만족해야 한다.

② 하드웨어적으로 임계 영역 문제를 해결하는 방법이 있다.

③ 세마포어 s의 p연산은 대기중인 프로세스를 깨우는 신호를 보내는 signal 동작으로 임계 영역에서 나오기 위한 연산을 나타낸다.

④ 모니터는 공유자원을 내부적으로 숨기고 공유자원에 접근하기 위한 인터페이스만을 제공함으로써 자원을 보호하고 프로세스 간에 동기화시킨다.

정답 3번

출제 포인트

임계 영역 문제의 해결 조건과 세마포어의 P·V 연산, 모니터의 역할을 구별하는 문제이다.

핵심 개념

임계 영역은 여러 프로세스가 동시에 실행하면 공유 데이터에 문제가 생길 수 있어 한 번에 하나만 실행하도록 보호해야 하는 코드 영역이다.
세마포어의 P 연산은 자원 사용을 요청하고 필요하면 대기하는 wait 동작이며, V 연산은 자원을 반납하고 대기 중인 프로세스를 깨울 수 있는 signal 동작이다.
모니터는 공유 데이터와 그 데이터에 접근하는 연산을 하나로 묶어 상호배제를 구조적으로 제공하는 고수준 동기화 기법이다.

선지별 해설

① O — 임계 영역 문제의 올바른 해결책은 상호배제, 진행, 제한된 대기를 만족해야 한다. 상호배제는 동시에 둘 이상이 임계 영역에 들어가지 못하게 하고, 진행은 진입할 프로세스의 결정을 불필요하게 미루지 않게 하며, 제한된 대기는 무한 대기를 막는다.
② O — Test-and-Set이나 Compare-and-Swap 같은 원자적 하드웨어 명령을 이용하여 임계 영역의 상호배제를 구현할 수 있다.
③ X — 틀린 표현은 ‘P연산은 대기 중인 프로세스를 깨우는 signal 동작으로 임계 영역에서 나오기 위한 연산’이라는 부분이다. P 연산은 세마포어 값을 감소시키고 자원을 사용할 수 없으면 대기하는 wait 동작으로, 임계 영역에 들어가기 전에 수행한다. 대기 중인 프로세스를 깨우고 임계 영역에서 나온 뒤 수행하는 signal 동작은 V 연산이다.
④ O — 모니터는 공유자원과 관련 연산을 내부에 캡슐화하고 정해진 인터페이스를 통해서만 접근하게 하며, 한 시점에 하나의 프로세스만 모니터 내부에서 활동하도록 하여 상호배제와 동기화를 지원한다.

NoteBase

오답 포인트

③은 P 연산과 V 연산의 기능을 서로 바꾸어 서술한 선지이다. ‘깨우는 신호’, ‘signal’, ‘임계 영역에서 나옴’은 모두 V 연산을 가리키며 P 연산의 설명이 아니다.

다시 나오면 이렇게 판단

세마포어 연산은 진입 전의 P(wait·감소·필요 시 대기)와 진출 후의 V(signal·증가·필요 시 깨움)로 구분한다. 임계 영역의 해결 조건은 상호배제·진행·제한된 대기인지 확인한다.

암기 포인트

P는 진입 전 wait, V는 진출 후 signal이다.

NoteBase

문 12. 프로세스와 쓰레드에 관한 다음 설명 중 가장 옳지 않은 것은?

① 같은 프로세스에 속한 쓰레드들은 가상 메모리를 공유한다.

② 프로세스가 스와핑(Swapping)되면 해당 프로세스에 속한 모든 쓰레드 역시 스와핑된다.

③ 쓰레드를 지원하기 위해서는 운영체제의 지원이 필수적이다.

④ 멀티코어 시스템에서는 쓰레드를 병렬로 처리하여 프로세스의 성능을 개선할 수 있다.

정답 3번

출제 포인트

프로세스와 쓰레드의 자원 공유 관계, 스와핑 단위, 사용자 수준 쓰레드와 커널 수준 쓰레드의 차이를 묻는 문제이다.

핵심 개념

프로세스는 독립된 가상 주소 공간과 자원을 가진 실행 단위이고, 쓰레드는 한 프로세스 안에서 실행 흐름을 나누는 단위이다. 같은 프로세스의 쓰레드는 주소 공간과 코드·데이터 등을 공유하지만, 각자 레지스터와 스택을 가진다. 사용자 수준 쓰레드는 사용자 영역의 쓰레드 라이브러리가 관리하므로 운영체제 커널이 쓰레드의 존재를 직접 알지 못해도 구현할 수 있다. 커널 수준 쓰레드는 운영체제가 직접 생성·관리·스케줄링한다.

선지별 해설

① O — 같은 프로세스에 속한 쓰레드는 그 프로세스의 가상 주소 공간을 공유하므로 코드와 전역 데이터 등에 함께 접근할 수 있다. 다만 각 쓰레드의 레지스터와 스택은 별도로 유지된다.
② O — 프로세스가 스와핑되면 그 프로세스의 주소 공간이 주기억장치에서 보조기억장치로 이동하므로, 그 주소 공간에 속한 모든 쓰레드도 실행할 수 없는 상태가 되어 함께 스와핑된 것으로 본다.
③ X — 틀린 표현은 '운영체제의 지원이 필수적이다'이다. 올바르게는 사용자 수준 쓰레드는 쓰레드 라이브러리만으로 구현할 수 있어 운영체제의 직접적인 쓰레드 지원이 필수적이지 않으며, 커널 수준 쓰레드는 운영체제의 지원이 필요하다.
④ O — 멀티코어 시스템에서는 운영체제가 여러 쓰레드를 서로 다른 코어에 배치할 수 있으므로 실제 병렬 실행을 통해 프로세스의 처리 성능을 높일 수 있다.

오답 포인트

③은 커널 수준 쓰레드의 특징을 모든 쓰레드에 일반화한 선지이다. '쓰레드'라는 말만 보고 운영체제 지원이 언제나 필요하다고 판단하면 사용자 수준 쓰레드를 놓치게 된다.

NoteBase

다시 나오면 이렇게 판단

쓰레드 구현 주체를 먼저 구분한다. 사용자 수준 쓰레드는 라이브러리가 관리하므로 커널의 직접 지원 없이도 구현할 수 있고, 커널 수준 쓰레드는 운영체제가 관리한다. 같은 프로세스의 쓰레드는 주소 공간을 공유하지만 실행 문맥인 레지스터와 스택은 각각 가진다.

암기 포인트

사용자 수준 쓰레드는 라이브러리 관리, 커널 수준 쓰레드는 운영체제 관리이다. 주소 공간은 공유하고 레지스터·스택은 개별 보유한다.

NoteBase

문 13. 다음 설명 중 가장 옳지 않은 것은?

① DIKW(Data-Information-Knowledge-Wisdom) 계층구조는 데이터가 지혜로 발전하는 과정을 나타낸다.

② 빅데이터의 3V는 데이터의 다양성, 속도, 데이터 양을 의미한다.

③ 빅데이터의 분석 기술로는 하둡, 텍스트 마이닝, 군집분석 등이 있다.

④ 키-값 데이터베이스는 NoSQL의 가장 단순한 형태이며, 키와 값의 쌍으로 구성된다.

정답 3번

출제 포인트

DIKW, 빅데이터의 3V, 빅데이터 분석 기법, NoSQL 데이터 모델의 기본 개념을 구별하는 문제이다.

핵심 개념

DIKW는 데이터가 정보, 지식, 지혜로 발전하는 계층 구조이다. 빅데이터의 3V는 규모(Volume), 속도(Velocity), 다양성(Variety)이다. 텍스트 마이닝과 군집분석은 데이터에서 의미나 패턴을 찾는 분석 기법이지만, 하둡은 대규모 데이터를 분산 저장하고 처리하기 위한 프레임워크이다. 키-값 데이터베이스는 고유한 키로 대응하는 값을 저장하고 조회하는 NoSQL 데이터 모델이다.

선지별 해설

① O — DIKW는 가공되지 않은 데이터가 맥락을 가진 정보, 이해된 지식, 판단에 활용되는 지혜로 발전하는 과정을 나타낸다.
② O — 빅데이터의 3V는 다양성(Variety), 속도(Velocity), 규모 또는 데이터 양(Volume)을 뜻한다.
③ X — 틀린 표현은 하둡을 텍스트 마이닝·군집분석과 같은 '분석 기술'로 묶은 부분이다. 하둡은 대규모 데이터를 분산 저장·처리하는 프레임워크이고, 텍스트 마이닝과 군집분석은 데이터의 의미나 패턴을 찾아내는 분석 기법이다. 올바르게는 '하둡은 빅데이터의 분산 저장·처리 기술이며, 텍스트 마이닝과 군집분석은 빅데이터 분석 기법이다'라고 해야 한다.
④ O — 키-값 데이터베이스는 각 키에 하나의 값을 대응시키는 단순한 구조의 NoSQL 데이터 모델이다.

오답 포인트

③은 하둡이 빅데이터 분야에서 널리 사용된다는 점을 이용해 '빅데이터 관련 기술'과 '빅데이터 분석 기법'을 혼동하게 한다. 하둡은 분석을 수행할 수 있는 저장·처리 기반이고, 텍스트 마이닝과 군집분석은 실제 분석 방법이다.

NoteBase

다시 나오면 이렇게 판단

기술의 역할을 기준으로 구별한다. 데이터를 보관하고 계산 작업을 분산 수행하면 저장·처리 기반이고, 데이터에서 의미·규칙·집단을 찾아내면 분석 기법이다.

암기 포인트

3V는 Volume·Velocity·Variety, 하둡은 분산 저장·처리, 텍스트 마이닝·군집분석은 분석 기법이다.

NoteBase

문 14. 데이터베이스에서 트랜잭션이 만족해야 하는 4가지 성질 중 “하나의 트랜잭션 수행 중 생성되는 연산의 중간 결과를 다른 트랜잭션이 접근할 수 없어야 한다”는 성질에 해당하는 것은?

① 원자성(Atomicity)
② 일관성(Consistency)
③ 고립성(Isolation)
④ 지속성(Durability)

정답 3번

출제 포인트

트랜잭션의 ACID 성질 중 다른 트랜잭션이 수행 중인 연산의 중간 결과에 접근하지 못하게 하는 성질을 구별하는 문제이다.

핵심 개념

원자성(Atomicity): 트랜잭션의 모든 연산이 전부 수행되거나 전혀 수행되지 않아야 하는 성질이다.
일관성(Consistency): 트랜잭션 수행 전후에 데이터베이스의 무결성 제약과 일관된 상태가 유지되어야 하는 성질이다.
고립성(Isolation): 여러 트랜잭션이 동시에 수행되더라도 각 트랜잭션이 다른 트랜잭션의 중간 결과를 볼 수 없도록 독립적으로 실행되는 성질이다.
지속성(Durability): 성공적으로 완료된 트랜잭션의 결과가 장애가 발생해도 영구적으로 보존되는 성질이다.

선지별 해설

① X — 원자성은 중간 결과의 접근 차단이 아니라 트랜잭션의 연산을 전부 수행하거나 전혀 수행하지 않는다는 성질이다. 틀린 표현인 ‘원자성’은 ‘고립성’으로 고쳐야 한다.
② X — 일관성은 중간 결과의 접근 차단이 아니라 트랜잭션 전후에 데이터베이스의 무결성과 일관된 상태를 유지하는 성질이다. 틀린 표현인 ‘일관성’은 ‘고립성’으로 고쳐야 한다.
③ O — 고립성은 수행 중인 트랜잭션의 중간 결과를 다른 트랜잭션이 접근할 수 없게 하여 트랜잭션들이 서로 독립적으로 실행되도록 한다.
④ X — 지속성은 중간 결과의 접근 차단이 아니라 완료된 트랜잭션의 결과가 장애 후에도 영구적으로 보존되는 성질이다. 틀린 표현인 ‘지속성’은 ‘고립성’으로 고쳐야 한다.

오답 포인트

‘하나의 트랜잭션’이라는 표현만 보고 원자성을 고르기 쉽다. 그러나 판단의 핵심은 ‘다른 트랜잭션이 중간 결과에 접근할 수 없다’는 부분이며, 이는 트랜잭션 사이의 간섭을 막는 고립성을 뜻한다.

NoteBase

다시 나오면 이렇게 판단

전부 수행 또는 전부 취소는 원자성, 무결성 유지와 정상 상태 전환은 일관성, 다른 트랜잭션의 중간 결과 차단은 고립성, 완료 결과의 영구 보존은 지속성으로 판단한다.

암기 포인트

ACID는 전부 아니면 전무(Atomicity), 규칙 유지(Consistency), 서로 격리(Isolation), 결과 영구 보존(Durability)이다.

NoteBase

문 15. 다음 설명 중 가장 옳지 않은 것은?

① 메모리 오버레이는 프로그램의 크기가 실제 메모리보다 클 때 전체 프로그램을 메모리에 가져오는 대신 적당한 크기로 잘라서 가져오는 기법이다.

② 스왑 영역에서 메모리로 데이터를 가져오는 작업을 스왑아웃이라 한다.

③ 메모리 분할방식은 가변분할방식과 고정분할방식으로 구분한다.

④ 가변분할방식은 프로세스의 크기에 맞춰 메모리를 할당하는 방법으로 세그멘테이션 메모리관리기법이라 한다.

정답 2번

출제 포인트

메모리 오버레이, 스와핑, 고정·가변 분할의 뜻을 구별하고 스왑인과 스왑아웃의 이동 방향을 판단하는 문제이다.

핵심 개념

메모리 오버레이: 주기억장치보다 큰 프로그램을 서로 동시에 필요하지 않은 부분으로 나누어 필요한 부분만 번갈아 적재하는 기법이다.
스와핑: 프로세스를 주기억장치와 보조기억장치의 스왑 영역 사이에서 이동시키는 기법이다. 스왑 영역에서 주기억장치로 가져오면 스왑인이고, 주기억장치에서 스왑 영역으로 내보내면 스왑아웃이다.
메모리 분할: 고정된 크기로 미리 나누는 고정분할과 프로세스 크기에 맞추어 나누는 가변분할로 구분한다. 이 문항의 분류에서는 가변분할을 가변 길이의 논리적 단위로 관리하는 세그멘테이션과 연결한다.

선지별 해설

① O — 메모리 오버레이는 프로그램 전체를 한꺼번에 적재할 수 없을 때 프로그램을 적당한 크기의 부분으로 나누고 필요한 부분만 메모리에 올리는 기법이다.
② X — 틀린 표현은 '스왑아웃'이다. 스왑 영역에서 주기억장치로 데이터를 가져오는 작업은 스왑인이고, 스왑아웃은 주기억장치의 데이터를 스왑 영역으로 내보내는 작업이다.
③ O — 메모리를 분할하여 할당하는 방식은 분할 크기를 미리 고정하는 고정분할방식과 프로세스 크기에 맞추어 정하는 가변분할방식으로 나뉜다.
④ O — 가변분할방식은 프로세스가 요구하는 크기에 맞추어 메모리를 할당한다. 이 문항이 사용하는 분류에서 가변 크기의 단위로 메모리를 관리하는 방식은 세그멘테이션 메모리 관리 기법에 해당한다.

오답 포인트

②는 스왑이 일어나는 사실 자체가 아니라 이동 방향의 명칭을 바꾸어 놓았다. 스왑 영역에서 메모리로 들어오는 방향을 스왑아웃으로 착각하기 쉽다.

NoteBase

다시 나오면 이렇게 판단

주기억장치를 기준으로 들어오면 스왑인, 밖의 스왑 영역으로 나가면 스왑아웃으로 판단한다. 분할 방식은 분할 크기가 미리 정해졌는지, 프로세스 크기에 따라 달라지는지를 확인한다.

암기 포인트

스왑 영역 → 주기억장치는 스왑인, 주기억장치 → 스왑 영역은 스왑아웃이다.

NoteBase

문 16. 다음 중 가상 머신(Virtual Machine)을 사용하는 이유로 가장 옳지 않은 것은?

① 여러 운영체제에서 실행되는 다양한 응용 프로그램들을 하나의 서버에서 실행할 수 있다.

② 서버의 성능을 증가시킬 수 있다.

③ 하나의 물리적 서버에 여러 개의 가상 머신을 실행할 수 있다.

④ 물리적 서버에 비하여 설치 시간을 단축시킬 수 있다.

정답 2번

출제 포인트

가상 머신을 사용하는 목적과 가상화가 제공하지 않는 효과를 구별하는 문제이다.

핵심 개념

가상 머신은 하나의 물리적 서버 자원을 논리적으로 분할하여 서로 독립된 운영체제와 응용 프로그램을 실행하는 환경이다. 핵심 효과는 서버 통합, 자원 활용률 향상, 환경 격리, 신속한 생성·배포이다. 다만 가상화 자체가 물리적 서버의 연산 성능을 높이는 것은 아니다.

선지별 해설

① O — 각 가상 머신에 서로 다른 운영체제를 설치할 수 있으므로, 운영체제가 다른 여러 응용 프로그램을 하나의 물리적 서버에서 실행할 수 있다.
② X — 틀린 표현은 '서버의 성능을 증가시킬 수 있다'이다. 가상화는 서버 자원의 활용률과 운영 효율을 높이지만 물리적 서버 자체의 처리 성능을 증가시키지는 않으며, 가상화 계층의 오버헤드가 발생할 수도 있다. 올바르게는 '서버 자원의 활용률과 관리 효율을 높일 수 있다'이다.
③ O — 한 대의 물리적 서버에서 CPU, 메모리, 저장장치 등의 자원을 나누어 여러 가상 머신을 동시에 실행할 수 있다.
④ O — 가상 머신은 미리 만든 이미지나 템플릿을 복제하여 생성할 수 있으므로 새 물리적 서버를 마련하고 운영체제를 설치하는 것보다 구축 시간을 단축할 수 있다.

오답 포인트

'자원 활용률 향상'을 '서버 자체의 성능 향상'으로 바꾸어 표현한 ②가 함정이다. 여러 작업을 한 서버에 모아 효율적으로 운영하는 것과 CPU·메모리의 물리적 성능이 커지는 것은 다른 개념이다.

다시 나오면 이렇게 판단

가상화의 장점은 통합·격리·자원 활용·신속한 배포에 해당하는지 판단한다. 물리 장비의 연산 능력 자체가 증가한다는 표현은 가상화의 직접적인 효과가 아니다.

NoteBase

암기 포인트

가상화는 성능 증가가 아니라 자원 활용률 증가이다.

NoteBase

문 17. 다음 전위표기(prefix) 식을 후위표기(postfix) 식으로 변환하였을 때 옳은 것은?

* + a b c

① a b c * +
② a + b * c
③ a b + c *
④ a b * c +

정답 3번

출제 포인트

전위표기식의 연산 구조를 파악한 뒤, 각 연산자를 두 피연산자 뒤로 옮겨 후위표기식으로 변환하는 과정을 묻는 문제이다.

올바른 순서

1. 전위표기식의 첫 연산자 *를 최상위 연산자로 본다.
2. *의 왼쪽 피연산자는 + a b이므로 a와 b를 먼저 +로 결합한다.
3. + a b를 후위표기로 바꾸면 a b +가 된다.
4. 오른쪽 피연산자 c를 적고 마지막에 최상위 연산자 *를 붙이면 a b + c *가 된다.

왜 이 순서: 안쪽 연산인 a+b의 결과가 바깥쪽 곱셈의 왼쪽 피연산자가 되므로 +가 *보다 먼저 배치된다.

NoteBase

선지별 해설

① X
- a b c * +는 b와 c를 먼저 곱한 뒤 a를 더하는 구조이다. 주어진 식에서는 a와 b를 먼저 더해야 한다.
- 틀린 부분: c *가 +보다 앞에 놓였다.
- 올바르게는: a b + c *이다.

② X
- a + b * c는 연산자가 피연산자 사이에 놓인 중위표기식이며, 요구된 후위표기식이 아니다.
- 틀린 부분: +와 *가 각 연산의 피연산자 뒤에 있지 않다.
- 올바르게는: a b + c *이다.

③ O
- a b +로 a와 b를 먼저 더하고, 그 결과와 c 뒤에 *를 놓아 전체 구조 (a+b)*c를 정확히 나타낸다.

④ X
- a b * c +는 a와 b를 먼저 곱한 뒤 c를 더하는 구조이다. 주어진 식의 최상위 연산자는 *이므로 후위표기에서도 *가 마지막에 와야 한다.
- 틀린 부분: *가 +보다 앞에 놓였다.
- 올바르게는: a b + c *이다.

오답 포인트

전위식의 연산자를 단순히 뒤로 보내면 피연산자의 묶음이 달라질 수 있다. 이 문항에서는 첫 기호 *가 최상위 연산자이고, 그 왼쪽의 + a b가 하나의 피연산자라는 점을 놓치면 ①이나 ④처럼 연산 순서를 바꾸게 된다.

다시 나오면 이렇게 판단

전위표기식은 앞에서부터 연산자마다 필요한 피연산자 두 개를 하나의 묶음으로 만든다. 각 묶음을 '왼쪽 피연산자 → 오른쪽 피연산자 → 연산자' 순서로 바꾸고, 안쪽 묶음부터 바깥쪽 묶음으로 결합하면 후위표기식이 된다.

암기 포인트

전위는 연산자가 앞, 후위는 연산자가 뒤이며, 괄호 구조는 그대로 유지된다.

NoteBase

문 18. 다음 중 데이터 수정을 위한 Update SQL 구문으로 올바른 것은?(단, 테이블의 속성은 2개로 구성됨)

① UPDATE INTO 테이블_이름[(속성이름1, 속성이름2)] SELECT 속성이름1, 속성이름2;

② UPDATE 테이블_이름 SET 속성이름1 = 값1, 속성이름2 = 값2;

③ UPDATE FROM 테이블_이름 WHERE 속성이름1 = 값1 AND 속성이름2 = 값2;

④ UPDATE 테이블_이름 VALUES('변경값1', '변경값2');

정답 2번

출제 포인트

UPDATE문의 기본 형식과 INSERT문에 쓰이는 키워드를 구별하는 문제이다.

핵심 개념

UPDATE문은 기존 행의 값을 변경하는 데이터 조작어이며 기본 형식은 'UPDATE 테이블_이름 SET 속성이름1 = 값1, 속성이름2 = 값2;'이다. WHERE절이 없으면 테이블의 모든 행이 수정되고, WHERE절을 붙이면 조건에 맞는 행만 수정된다. INSERT문은 새 행을 추가하며 INSERT INTO와 VALUES 또는 SELECT를 사용한다.

선지별 해설

① X — 'UPDATE INTO'와 그 뒤의 'SELECT'가 잘못되었다. INSERT문이라면 'INSERT INTO 테이블_이름(속성이름1, 속성이름2) SELECT …'와 같이 쓰지만, 기존 데이터를 수정하려면 'UPDATE 테이블_이름 SET 속성이름1 = 값1, 속성이름2 = 값2;'로 고쳐야 한다.
② O — 'UPDATE 테이블_이름 SET 속성이름1 = 값1, 속성이름2 = 값2;'는 두 속성을 수정하는 올바른 UPDATE 구문이다. WHERE절이 없으므로 모든 행의 두 속성값을 변경한다.
③ X — 'UPDATE FROM'이 잘못되었고, 변경 내용을 지정하는 SET절도 없다. 'UPDATE 테이블_이름 SET 속성이름1 = 값1, 속성이름2 = 값2 WHERE 조건;'처럼 고쳐야 한다.
④ X — 'VALUES'는 UPDATE문에서 변경값을 지정하는 키워드가 아니다. 'UPDATE 테이블_이름 SET 속성이름1 = '변경값1', 속성이름2 = '변경값2';'처럼 VALUES를 SET과 각 속성의 대입식으로 고쳐야 한다.

오답 포인트

①의 INTO·SELECT와 ④의 VALUES는 데이터 추가에 사용하는 INSERT문의 형태를 UPDATE문에 섞어 놓은 함정이다. ③은 WHERE 조건만 제시하고 정작 무엇을 바꿀지 나타내는 SET절을 빠뜨렸다.

다시 나오면 이렇게 판단

기존 행 수정 문제에서는 먼저 'UPDATE 테이블명 SET 열 = 값'의 순서가 있는지 확인한다. INTO나 VALUES가 중심이면 INSERT문 형태인지 의심하고, WHERE절은 수정 대상 행을 제한하는 선택 요소로 판단한다.

NoteBase

암기 포인트

수정은 UPDATE–SET, 추가는 INSERT INTO–VALUES이다.

NoteBase

문 19. 다음 설명 중 가장 옳지 않은 것은?

① 하나의 릴레이션에서 속성의 전체 개수를 카디널리티라 한다.

② 릴레이션 스키마는 릴레이션의 이름과 릴레이션에 포함된 모든 속성의 이름을 정의하는 릴레이션의 논리적 구조를 의미한다.

③ 릴레이션의 특성중 튜플(Tuple)의 유일성과 속성의 무순서 특성도 존재한다.

④ 릴레이션의 특성중 속성의 원자성은 속성값으로 원자값만을 사용할 수 있다라는 의미이다.

정답 1번

출제 포인트

릴레이션의 차수와 카디널리티를 구별하고, 릴레이션 스키마와 릴레이션의 기본 특성을 이해하는지를 묻는 문제이다.

핵심 개념

차수(degree)는 릴레이션을 구성하는 속성의 개수이고, 카디널리티(cardinality)는 릴레이션에 포함된 튜플의 개수이다. 릴레이션 스키마는 릴레이션의 이름과 속성 등 논리적 구조를 정의하며, 릴레이션에서는 튜플이 중복되지 않고 속성의 순서에 의미가 없으며 각 속성값은 원자값이어야 한다.

선지별 해설

① X
속성의 전체 개수를 카디널리티라고 한 부분이 틀렸다. 올바르게는 속성의 전체 개수를 차수라고 하고, 튜플의 전체 개수를 카디널리티라고 한다.

② O
릴레이션 스키마는 릴레이션의 이름과 그 릴레이션을 구성하는 속성의 이름 등 릴레이션의 논리적 구조를 정의한다.

③ O
릴레이션에서는 동일한 튜플이 중복될 수 없으므로 튜플의 유일성이 성립하며, 속성을 나열한 순서는 릴레이션의 의미에 영향을 주지 않는다.

④ O
속성의 원자성은 하나의 속성값이 더 이상 나눌 필요가 없는 단일 값, 즉 원자값이어야 한다는 뜻이다.

NoteBase

오답 포인트

①은 속성의 개수를 뜻하는 차수와 튜플의 개수를 뜻하는 카디널리티를 서로 바꾸어 제시한 선지이다. 두 용어를 행과 열에 연결하지 못하면 쉽게 혼동한다.

다시 나오면 이렇게 판단

릴레이션을 표처럼 생각하여 열에 해당하는 속성의 수는 차수, 행에 해당하는 튜플의 수는 카디널리티로 판단한다. 스키마는 구조의 정의이고, 릴레이션의 기본 특성은 튜플의 유일성, 튜플과 속성의 무순서성, 속성값의 원자성으로 구분한다.

암기 포인트

열(속성)의 수는 차수, 행(튜플)의 수는 카디널리티이다.

NoteBase

문 20. 다음 C 언어의 배열에서 a[2][1]의 값은?

```
int a[4][3] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
```

① 2

② 3

③ 7

④ 8

정답 4번

출제 포인트

C 언어의 2차원 배열이 행 우선 순서로 초기화되고, 배열 인덱스는 0부터 시작한다는 점을 적용하는 문제이다.

선지별 해설

① X
2는 a[0][1]의 값이다. a[2][1]의 값으로는 틀리며, 올바른 값은 8이다.

② X
3은 a[0][2]의 값이다. a[2][1]의 값으로는 틀리며, 올바른 값은 8이다.

③ X
7은 a[2][0]의 값이다. 열 인덱스 1이 아니라 0에 해당하므로 틀리며, a[2][1]의 올바른 값은 8이다.

④ O
각 행에 원소가 3개씩 들어가므로 a[2]는 세 번째 행 {7, 8, 9}이다. 그중 열 인덱스 1은 두 번째 원소이므로 a[2][1]은 8이다.

오답 포인트

a[2][1]을 두 번째 행의 첫 번째 원소로 읽는 것이 함정이다. C 언어의 인덱스는 0부터 시작하므로 행 인덱스 2는 세 번째 행, 열 인덱스 1은 두 번째 열을 뜻한다.

다시 나오면 이렇게 판단

먼저 한 행의 원소 수를 확인해 초기값을 행별로 묶고, 행과 열의 인덱스를 각각 0부터 센다. 이 배열은 {1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12}로 배치되므로 a[2][1]은 8이다.

NoteBase

암기 포인트

C 배열은 0번부터 시작하며, 2차원 배열의 연속 초기값은 행을 먼저 채운다.

NoteBase

문 21. 하드디스크에서 헤드가 읽기/쓰기 요청을 처리하기 위해 데이터가 존재하는 트랙(또는 실린더)으로 움직이는데 걸리는 시간을 뜻하는 용어는?

① 탐색시간(seek time)
② 회전지연시간(rotational latency)
③ 전송시간(transfer time)
④ 접근시간(access time)

정답 1번

출제 포인트

하드디스크의 접근 과정에서 헤드 이동, 원판 회전 대기, 실제 데이터 전송에 각각 걸리는 시간을 구별하는 문제이다.

핵심 개념

탐색시간은 헤드를 원하는 트랙 또는 실린더로 이동시키는 시간이다. 회전지연시간은 원하는 섹터가 헤드 아래에 올 때까지 디스크 원판의 회전을 기다리는 시간이다. 전송시간은 위치를 찾은 뒤 데이터를 실제로 읽거나 쓰는 시간이다. 접근시간은 이러한 단계에 걸리는 시간을 합한 포괄적인 시간이다.

선지별 해설

① O — 탐색시간(seek time)은 헤드가 데이터가 있는 트랙 또는 실린더로 이동하는 데 걸리는 시간이므로 문제의 정의와 일치한다.
② X — '회전지연시간'이 틀린 표현이다. 회전지연시간은 헤드 이동 시간이 아니라 원하는 섹터가 헤드 아래로 회전해 올 때까지 기다리는 시간이다. 제시된 정의에 맞는 용어는 탐색시간이다.
③ X — '전송시간'이 틀린 표현이다. 전송시간은 헤드 위치가 정해진 뒤 데이터를 실제로 읽거나 쓰며 전송하는 시간이다. 헤드가 해당 트랙으로 이동하는 시간은 탐색시간이다.
④ X — '접근시간'이 틀린 표현이다. 접근시간은 탐색시간과 회전지연시간 등 데이터에 접근하는 데 필요한 시간을 합한 포괄적 개념이다. 헤드가 트랙 또는 실린더로 이동하는 단계만을 가리키는 용어는 탐색시간이다.

오답 포인트

④의 '접근시간'은 탐색시간을 포함하는 상위 개념이어서 혼동하기 쉽다. 문제에서 '헤드가 트랙 또는 실린더로 움직인다'고 했으므로 전체 접근 시간이 아니라 탐색시간을 골라야 한다.

다시 나오면 이렇게 판단

헤드가 트랙으로 이동하면 탐색시간, 섹터가 돌아오기를 기다리면 회전지연시간, 데이터를 실제로 주고받으면 전송시간, 여러 단계를 합쳐 말하면 접근시간으로 판단한다.

NoteBase

암기 포인트

트랙까지 찾는 시간은 탐색, 섹터를 기다리는 시간은 회전지연, 데이터를 옮기는 시간은 전송이다.

NoteBase

문 22. 다음 중 여러 대의 컴퓨터가 하나의 인터넷 연결선을 공유하기 위해 사용하는 기술로 가장 올바른 것은?

① DHCP

② DNS

③ NAT

④ SMTP

정답 3번

출제 포인트

여러 컴퓨터가 하나의 공인 IP 주소와 인터넷 연결을 공유하게 하는 NAT의 기능을 다른 네트워크 서비스와 구별하는 문제이다.

핵심 개념

NAT(Network Address Translation)는 내부 사설 IP 주소와 외부 공인 IP 주소를 변환하는 기술이다. 가정이나 사무실의 공유기는 이 기능을 이용해 여러 기기가 하나의 공인 IP 주소로 인터넷에 접속하게 한다. 여러 내부 연결을 포트 번호로 구분하는 방식은 더 정확히 NAPT 또는 PAT라고 하지만, 일반적으로 NAT의 기능에 포함하여 설명한다.

선지별 해설

① X — DHCP는 네트워크의 단말에 IP 주소, 서브넷 마스크, 기본 게이트웨이 등의 설정을 자동으로 할당한다. '인터넷 연결선을 공유하는 기술'이라는 표현이 틀렸으며, 올바르게는 '단말에 네트워크 설정을 자동 할당하는 기술'이다.
② X — DNS는 도메인 이름을 해당 IP 주소로 변환한다. '인터넷 연결선을 공유하는 기술'이라는 표현이 틀렸으며, 올바르게는 '도메인 이름과 IP 주소를 대응시키는 이름 해석 서비스'이다.
③ O — NAT는 내부의 여러 사설 IP 주소를 외부의 공인 IP 주소로 변환하므로 여러 컴퓨터가 하나의 공인 IP 주소와 인터넷 연결을 공유할 수 있게 한다.
④ X — SMTP는 전자우편을 발송하거나 메일 서버 사이에서 전송할 때 사용하는 프로토콜이다. '인터넷 연결선을 공유하는 기술'이라는 표현이 틀렸으며, 올바르게는 '전자우편 전송 프로토콜'이다.

오답 포인트

DHCP도 공유기에서 자주 함께 작동하므로 NAT와 혼동하기 쉽다. DHCP는 내부 기기에 주소 설정을 나누어 주고, NAT는 내부 주소와 외부 주소를 변환해 인터넷 연결을 공유하게 한다는 차이가 핵심이다.

NoteBase

다시 나오면 이렇게 판단

‘여러 기기·사설 IP·하나의 공인 IP·인터넷 공유’가 함께 제시되면 NAT를 고른다. ‘IP 설정 자동 할당’은 DHCP, ‘도메인 이름 변환’은 DNS, ‘메일 전송’은 SMTP로 판단한다.

암기 포인트

주소 자동 할당은 DHCP, 이름 해석은 DNS, 주소 변환과 인터넷 공유는 NAT, 메일 전송은 SMTP이다.

NoteBase

문 23. 정적 테스트 방법은 비정형 방법과 정형 방법으로 나눌 수 있다. 다음 중 비정형 방법에 해당하지 않는 것은?

① 개별검토
② 소프트웨어검사
③ 검토회의
④ 모형검증기법

정답 4번

출제 포인트

정적 테스트의 비정형 방법과 정형 방법을 구분하는 문제이다.

핵심 개념

정적 테스트는 프로그램을 실행하지 않고 문서나 소스 코드 등을 검토하여 결함을 찾는 방법이다.
비정형 방법은 사람의 검토와 경험을 중심으로 결함을 찾으며, 개별검토·소프트웨어검사·검토회의 등이 해당한다.
정형 방법은 수학적 모델과 논리적 절차를 이용하여 시스템의 성질을 검증하며, 모형검증기법 등이 해당한다.

선지별 해설

① O — 개별검토는 개발자나 검토자가 산출물을 직접 읽고 오류를 찾는 비정형 방법이다.
② O — 소프트웨어검사는 사람이 코드나 문서를 체계적으로 살펴 결함을 찾는 방법이므로 비정형 방법에 해당한다. 절차가 체계적이라는 이유만으로 수학적 정형 방법이 되는 것은 아니다.
③ O — 검토회의는 여러 검토자가 회의를 통해 오류와 문제점을 발견하는 사람 중심의 비정형 방법이다.
④ X — 모형검증기법은 시스템을 상태 전이 모형 등으로 표현하고 요구한 성질이 성립하는지를 논리적으로 검증하는 정형 방법이다. 틀린 표현은 '비정형 방법'이며, 올바르게는 '정형 방법'이다.

오답 포인트

소프트웨어검사가 정해진 절차에 따라 수행된다는 점 때문에 이를 정형 방법으로 오인하기 쉽다. 여기서 정형 방법은 단순히 절차가 체계적이라는 뜻이 아니라 수학적 모델과 논리를 사용하는 검증을 뜻한다.

다시 나오면 이렇게 판단

사람이 문서나 코드를 읽고 검토하면 비정형 방법으로, 상태 모형·논리식·수학적 증명을 이용하면 정형 방법으로 판단한다.

NoteBase

암기 포인트

사람의 검토는 비정형, 수학적 모형의 검증은 정형이다.

NoteBase

문 24. 다음 중 TCP 기반의 프로토콜이 아닌 것은?

① HTTP (Hypertext Transfer Protocol)

② FTP (File Transfer Protocol)

③ DNS (Domain Name System)

④ SMTP (Simple Mail Transfer Protocol)

정답 3번

출제 포인트

응용 계층 프로토콜이 주로 TCP와 UDP 중 어느 전송 계층 프로토콜을 사용하는지 구분하는 문제이다.

핵심 개념

TCP는 연결을 설정하고 순서 제어·재전송·오류 제어를 통해 신뢰성 있는 전송을 제공한다. UDP는 연결 설정 없이 데이터를 빠르게 전달하며, 재전송이나 순서 보장을 기본으로 제공하지 않는다. HTTP·FTP·SMTP는 일반적으로 TCP를 사용하고, DNS의 일반적인 질의와 응답은 주로 UDP를 사용한다. 다만 DNS도 영역 전송이나 응답이 큰 경우에는 TCP를 사용할 수 있다.

선지별 해설

① O — HTTP/1.1과 HTTP/2는 웹 문서와 자원을 신뢰성 있게 전송하기 위해 TCP를 사용한다.
② O — FTP는 제어 연결과 데이터 전송에 TCP를 사용한다.
③ X — 틀린 부분은 DNS를 TCP 기반 프로토콜로 분류하는 것이다. DNS의 일반적인 이름 조회는 주로 UDP를 사용하며, TCP는 영역 전송이나 큰 응답 등 필요한 경우에 사용한다. 따라서 올바르게는 'DNS는 일반적인 질의·응답에서 주로 UDP를 사용하는 프로토콜이다'이다.
④ O — SMTP는 전자우편을 안정적으로 전송하기 위해 TCP를 사용한다.

오답 포인트

DNS도 특정 상황에서는 TCP를 사용한다는 사실 때문에 ③을 놓치기 쉽다. 이 문제는 예외적 사용 가능성이 아니라 각 프로토콜의 일반적인 전송 방식을 묻는다. HTTP·FTP·SMTP는 TCP, 일반적인 DNS 질의·응답은 UDP로 구분한다.

다시 나오면 이렇게 판단

웹 전송·파일 전송·메일 전송처럼 신뢰성과 순서 보장이 중요한 HTTP·FTP·SMTP는 TCP로 판단한다. 짧은 이름 조회처럼 빠른 요청·응답이 중심인 DNS는 우선 UDP로 판단하되, 영역 전송이나 큰 응답에서는 TCP도 사용한다는 예외를 함께 구분한다.

NoteBase

암기 포인트

HTTP·FTP·SMTP는 TCP, DNS의 일반 조회는 UDP이다.

NoteBase

문 25. 다음 중 인터프리터(Interpreter)에 해당하지 않는 것은?

① 자바 가상 머신(Java Virtual Machine)

② 자바(Java) 컴파일러

③ 파이썬(Python) 인터프리터

④ PHP 인터프리터

정답 2번

출제 포인트

인터프리터처럼 코드를 실행하는 구성 요소와 소스 코드를 다른 형태로 번역하는 컴파일러를 구별하는 문제이다.

핵심 개념

인터프리터는 프로그램을 실행하면서 명령을 해석하여 처리하는 번역·실행 방식이다. 컴파일러는 실행 전에 소스 프로그램을 목적 코드나 중간 코드로 번역한다. 자바에서는 자바 컴파일러가 소스 코드를 바이트코드로 번역하고, 자바 가상 머신이 그 바이트코드를 해석하거나 JIT 컴파일하여 실행한다.

선지별 해설

① X — 자바 가상 머신은 자바 바이트코드를 해석하여 실행하는 기능을 수행하므로 이 문항의 분류에서는 인터프리터에 해당한다. '자바 가상 머신은 인터프리터에 해당하지 않는다'는 판단이 틀리며, 올바르게는 자바 바이트코드의 실행 환경으로서 해석 실행 기능을 제공한다고 해야 한다.
② O — 자바 컴파일러는 자바 소스 코드를 바이트코드로 번역하는 컴파일러이므로 인터프리터에 해당하지 않는다.
③ X — 파이썬 인터프리터는 파이썬 프로그램을 해석·실행하므로 인터프리터에 해당한다. '파이썬 인터프리터는 인터프리터에 해당하지 않는다'는 판단이 틀리며, 올바르게는 파이썬 코드를 실행하는 인터프리터라고 해야 한다.
④ X — PHP 인터프리터는 PHP 코드를 해석·실행하므로 인터프리터에 해당한다. 'PHP 인터프리터는 인터프리터에 해당하지 않는다'는 판단이 틀리며, 올바르게는 PHP 코드를 실행하는 인터프리터라고 해야 한다.

오답 포인트

자바 프로그램의 실행 과정에 컴파일과 해석이 모두 등장한다는 점이 함정이다. 자바 컴파일러는 소스 코드를 바이트코드로 번역하고, 자바 가상 머신은 생성된 바이트코드를 실행한다. 따라서 '자바'라는 공통 명칭이 아니라 각 구성 요소의 역할로 판단해야 한다.

NoteBase

다시 나오면 이렇게 판단

선지의 구성 요소가 소스 코드를 다른 코드로 미리 번역하면 컴파일러이고, 프로그램 실행 과정에서 코드를 해석하여 처리하면 인터프리터로 판단한다.

암기 포인트

자바 컴파일러는 소스 코드에서 바이트코드로 번역하고, JVM은 바이트코드를 실행한다.

NoteBase